Computer Animation II

Orientation interpolation

Dynamics

Some slides courtesy of Leonard McMillan and Jovan Popovic

Review from Thursday

Interpolation

- Splines

Articulated bodies

- Forward kinematics
- Inverse Kinematics
- Optimization
  - Gradient descent
    - Following the steepest slope
  - Lagrangian multipliers
    - Turn optimization with constraints into no constraints

Debugging

Debug all sub-parts as you write them

Print as much information as possible

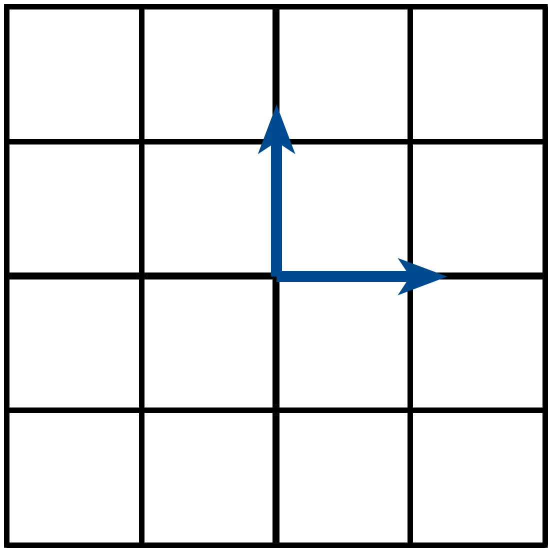
Use simple test cases

When stuck, use step-by-step debugging

Grid acceleration

Debug all these steps:

- Sphere rasterization
- Ray initialization
- Marching
- Object insertion
- Acceleration



Final project

First brainstorming session on Thursday

Groups of three

Proposal due Monday 10/27

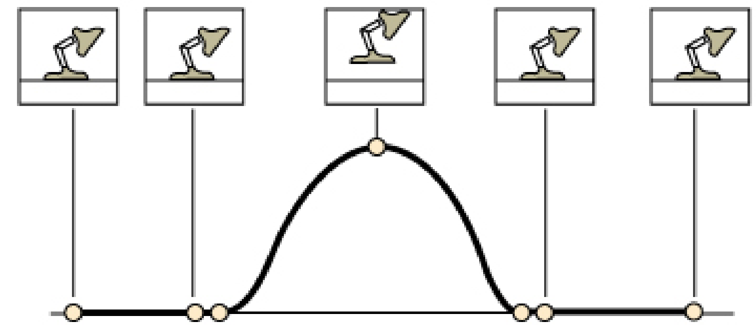
- A couple of pages
- Goals
- Progression

Appointment with staff

Computer-Assisted Animation

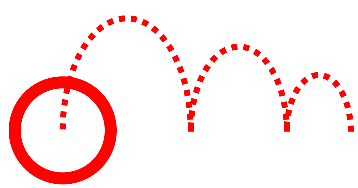
Keyframing
- automate the inbetweening
- good control
- less tedious
- creating a good animation still requires considerable skill and talent

Procedural animation
- describes the motion algorithmically
- express animation as a function of small number of parameteres
- Example: a clock with second, minute and hour hands
  - hands should rotate together
  - express the clock motions in terms of a "seconds" variable
  - the clock is animated by varying the seconds parameter
- Example 2: A bouncing ball
  - $Abs(\sin(\omega t+\theta_0))*e^{-kt}$



ACM © 1987 "Principles of traditional animation applied to 3D computer animation"

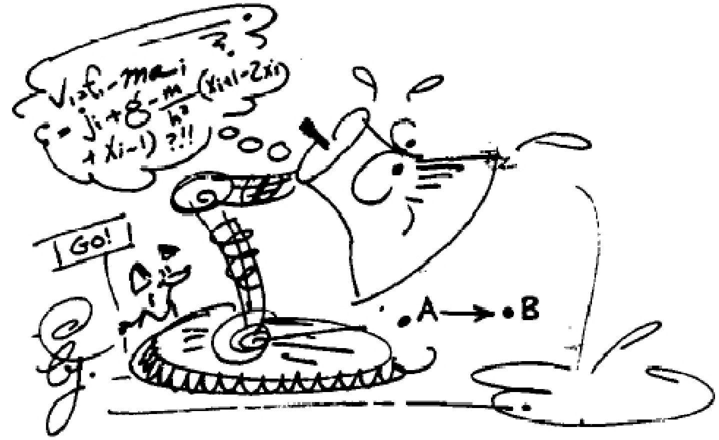
Image adapted from:
Lasseter, John. *Principles of Traditional Animation applied to 3D Computer Animation*, *ACM SIGGRAPH Computer Graphics*. 21, no.4 (July 1987): 35-44, applied to 3D computer animation.



Computer-Assisted Animation

Physically Based Animation

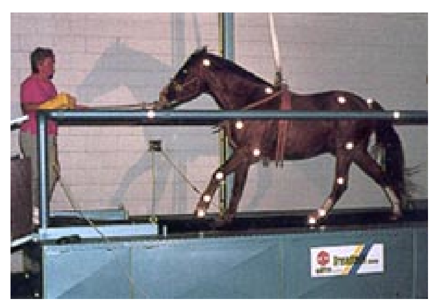
- Assign physical properties to objects (masses, forces, inertial properties)
- Simulate physics by solving equations
- Realistic but difficult to control



ACM© 1988 “Spacetime Constraints”

Motion Capture

- Captures style, subtle nuances and realism
- You must observe someone do something



Overview

Interpolation of rotations, quaternions

- Euler angles
- Quaternions

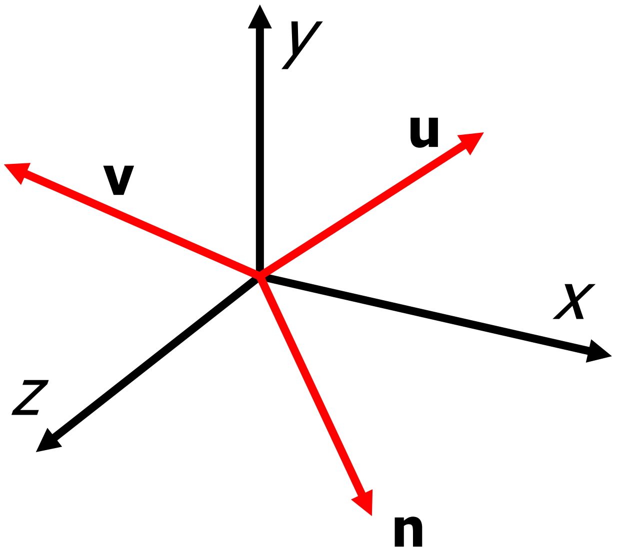
Dynamics

- Particles
- Rigid body
- Deformable objects

Interpolating Orientations in 3-D

Rotation matrices

Given rotation matrices M_i and time t_i, find $M(t)$ such that $M(t_i)=M_i$.



$$M = \begin{pmatrix} u_x & u_y & u_z \\ v_x & v_y & v_z \\ n_x & n_y & n_z \end{pmatrix}$$

Flawed Solution

Interpolate each entry independently

Example: M_0 is identity and M_1 is 90^o around x-axis

$$\text{Interpolate}\left(\begin{bmatrix}1&0&0\\0&1&0\\0&0&1\end{bmatrix},\begin{bmatrix}1&0&0\\0&0&1\\0&-1&0\end{bmatrix}\right)=\begin{bmatrix}1&0&0\\0&0.5&0.5\\0&-0.5&0.5\end{bmatrix}$$

Is the result a rotation matrix?

NO:

For example, RR^T does not equal identity.

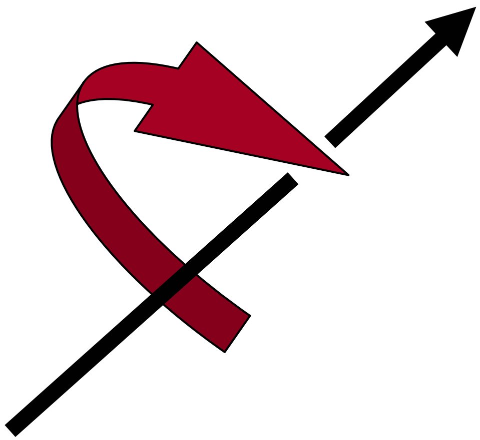
This interpolation does not preserve rigidity (angles and lengths)

3D rotations

How many degrees of freedom for 3D orientations?

3 degrees of freedom:

e.g. direction of rotation and angle



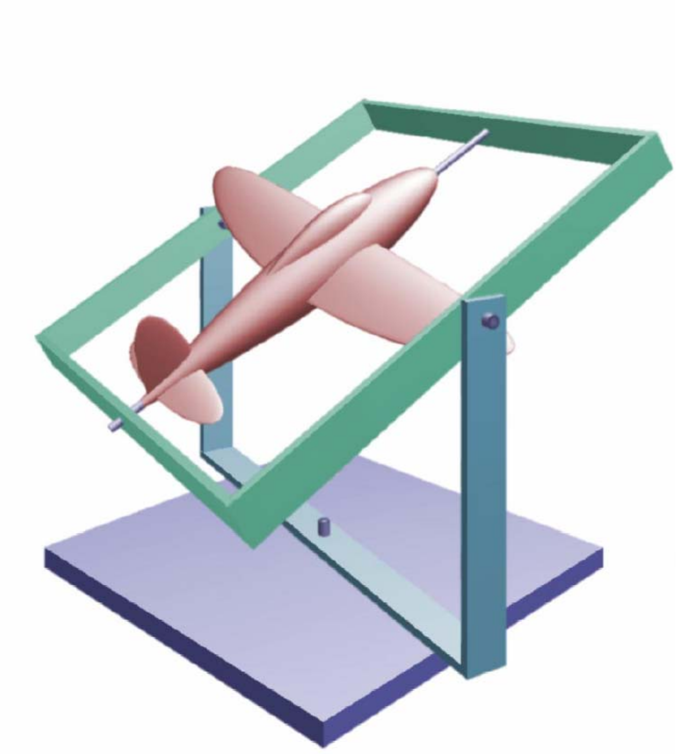
Or 3 Euler angles

Euler Angles

An Euler angle is a rotation about a single axis. Any orientation can be described composing three rotation around each coordinate axis.

Roll, pitch and yaw

Courtesy of Gernot Hoffmann. Used with permission.



http://www.fho-emden.de/~hoffmann/gimbal09082002.pdf

Interpolating Euler Angles

Natural orientation representation:

3 angles for 3 degrees of freedom

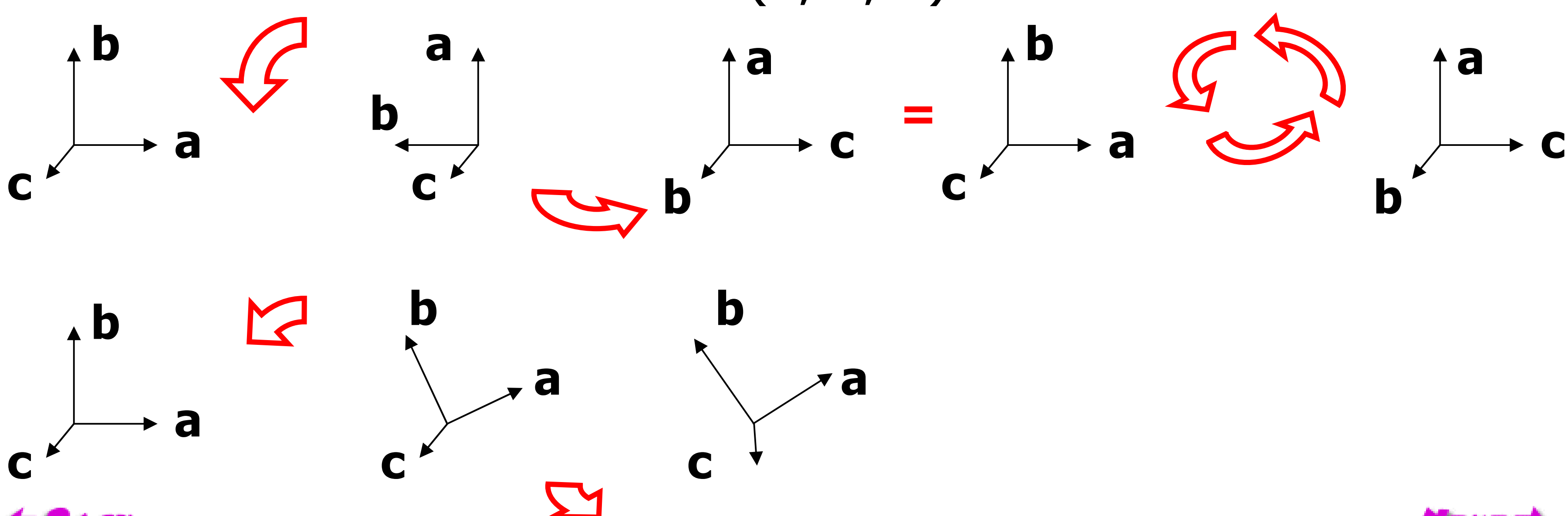
Unnatural interpolation:

A rotation of 90^o first around Z and then around Y

= 120^o around (1, 1, 1).

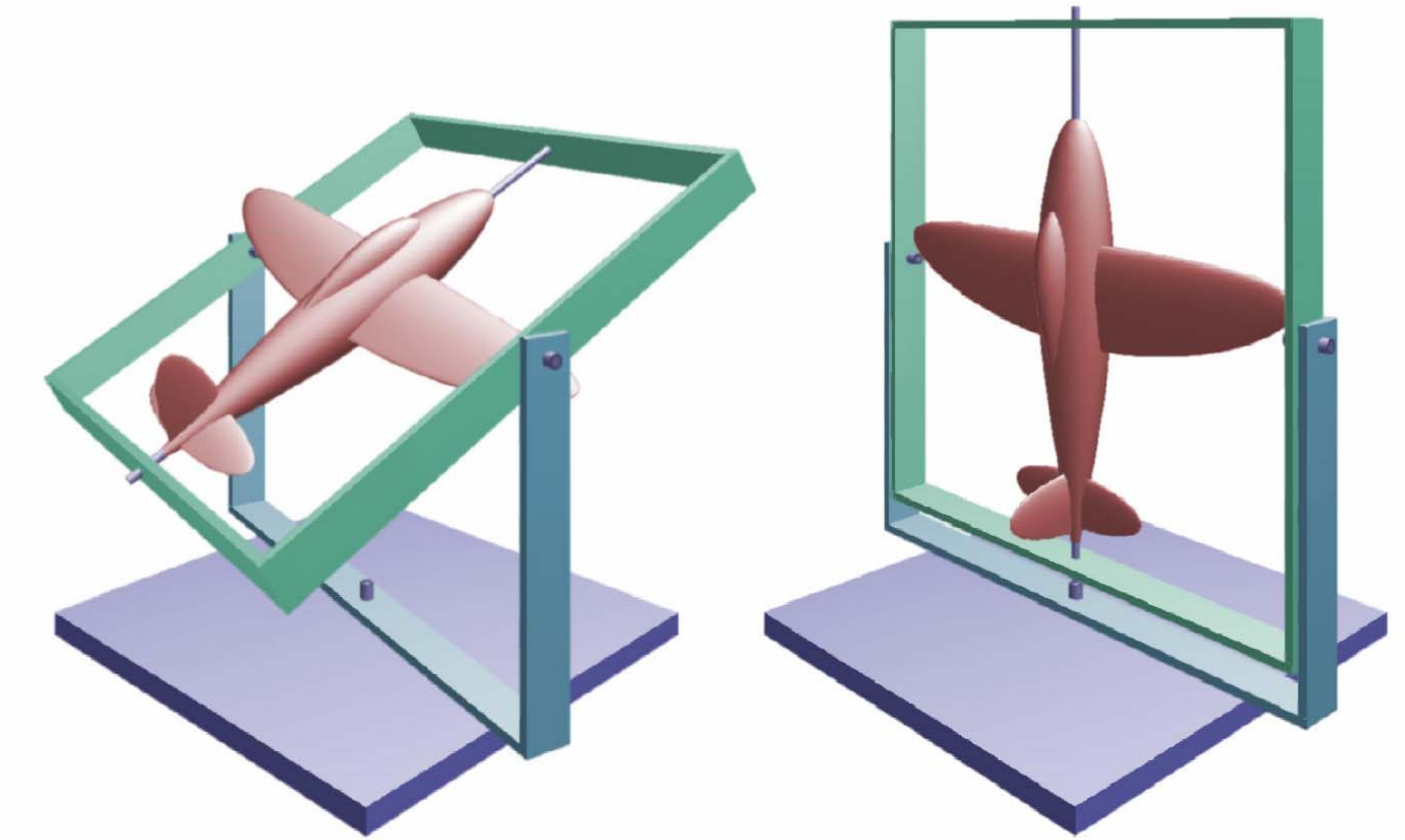
But 30^o around Z then Y

differs from 40^o around (1, 1, 1).



Gimbal lock

Gimbal lock: two or more axis align resulting in a loss of rotation degrees of freedom.



http://www.fho-emden.de/~hoffmann/gimbal09082002.pdf

Demo

By Sobeit void

from
http://www.gamedev.net/reference/programming/features/qpowers/page7.asp

See also
http://www.cgl.uwaterloo.ca/GALLERY/image_html/gimbal.jpg.html

Euler angles in the real world

Apollo inertial measurement unit

To prevent lock, they had to add a fourth Gimbal!

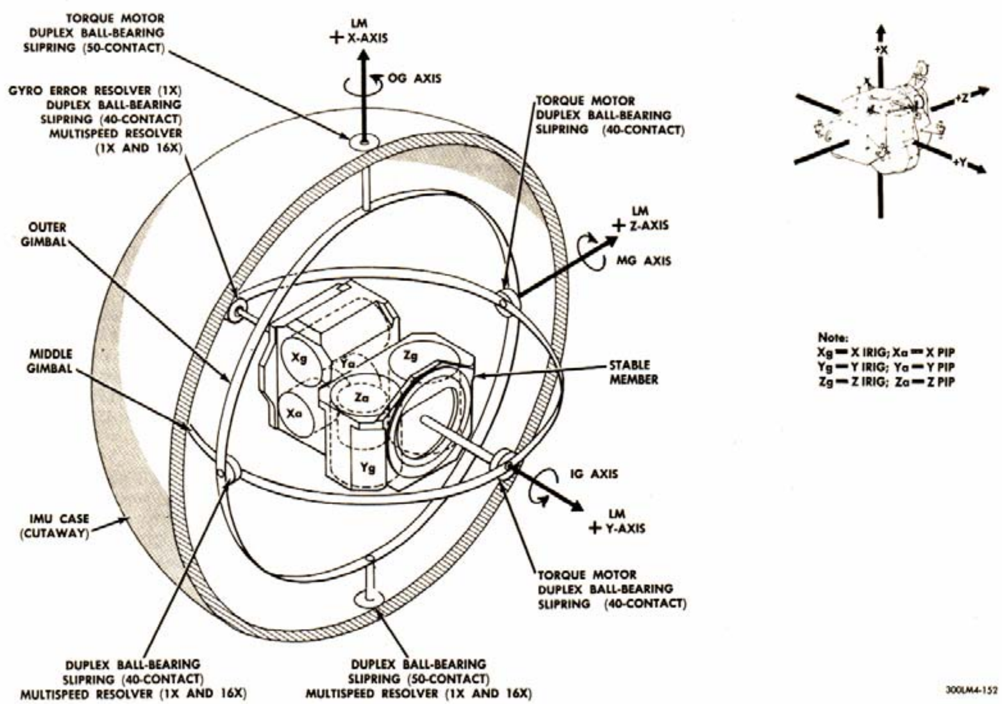
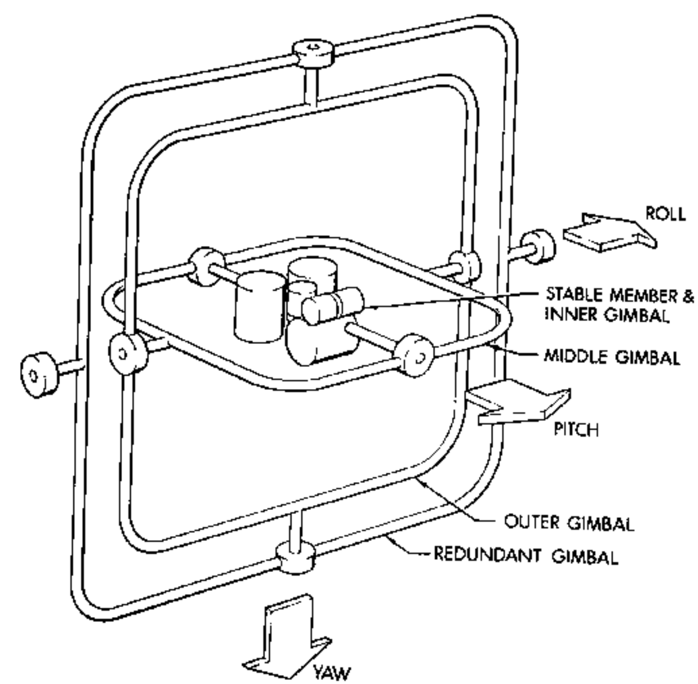


Figure 2.1-24. IMU Gimbal Assembly



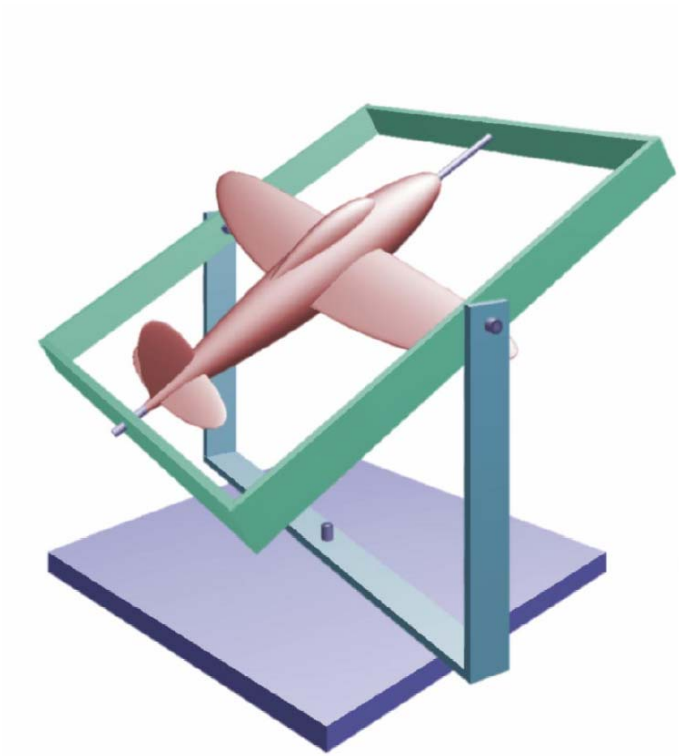
http://www.hq.nasa.gov/office/pao/History/alsj/gimbals.html

Recap: Euler angles

3 angles along 3 axis

Poor interpolation, lock

But used in flight simulation, etc. because natural



http://www.fho-emden.de/~hoffmann/gimbal09082002.pdf

Overview

Interpolation of rotations, quaternions

- Euler angles
- Quaternions

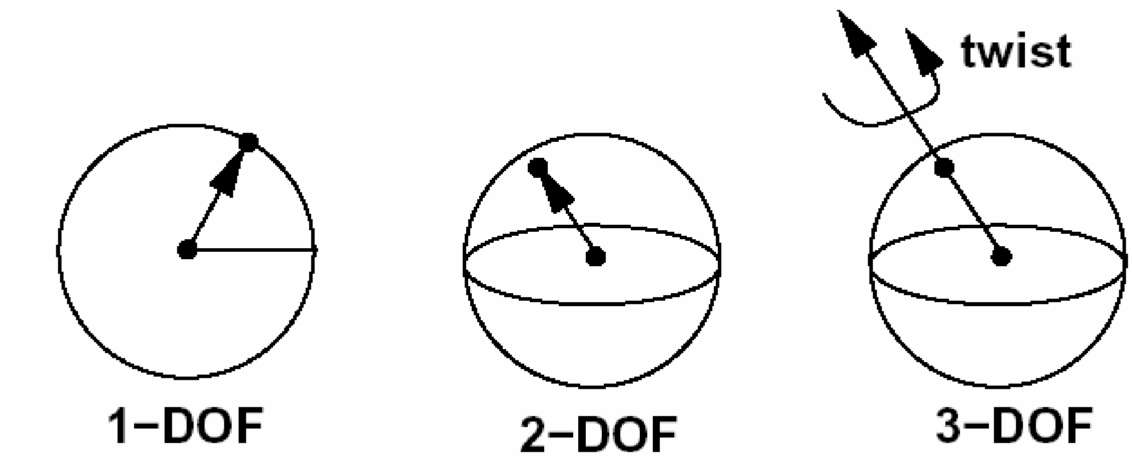
Dynamics

- Particles
- Rigid body
- Deformable objects

Solution: Quaternion Interpolation

Interpolate orientation on the unit sphere

By analogy: 1-, 2-, 3-DOF rotations as constrained points on 1, 2, 3-spheres



1D sphere and complex plane

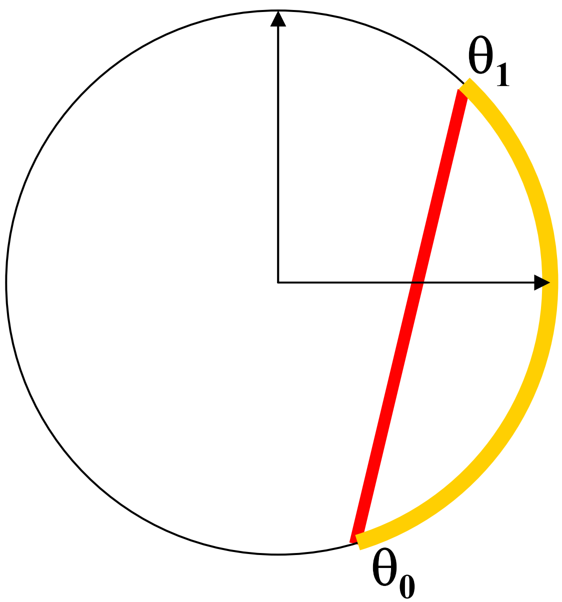
Interpolate orientation in 2D

1 angle

- But messy because modulo 2π

Use interpolation in (complex) 2D plane

Orientation = complex argument of the number

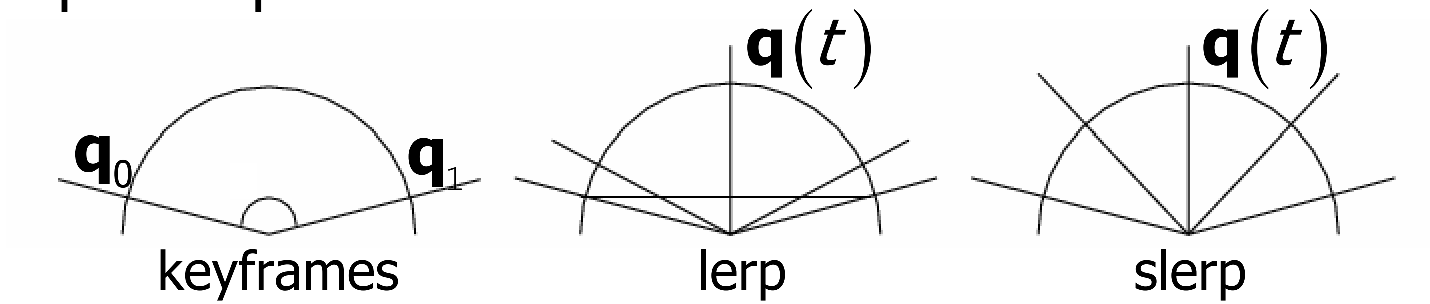


Velocity issue & slerp

linear int**erp**olation (lerp) in the plane interpolates the straight line between the two orientations and not their spherical distance.

$$\text{lerp}\left(\mathbf{q}_0,\mathbf{q}_1,t\right)=\mathbf{q}(t)=\mathbf{q}_0\left(1-t\right)+\mathbf{q}_1 t$$

=> The interpolated motion does not have uniform velocity: it may speed up too much:



Solution: Spherical **l**inear int**erp**olation (slerp): interpolate along the arc lines by adding a sine term.

$$\text{slerp}\left(\mathbf{q}_0,\mathbf{q}_1,t\right)=\mathbf{q}(t)=\frac{\mathbf{q}_0\sin\left(\left(1-t\right)\omega\right)+\mathbf{q}_1\sin\left(t\omega\right)}{\sin\left(\omega\right)},$$

where $\omega=\cos^{-1}\left(\mathbf{q}_0\bullet\mathbf{q}_1\right)$

2-angle orientation

Embed 2-sphere in 3D

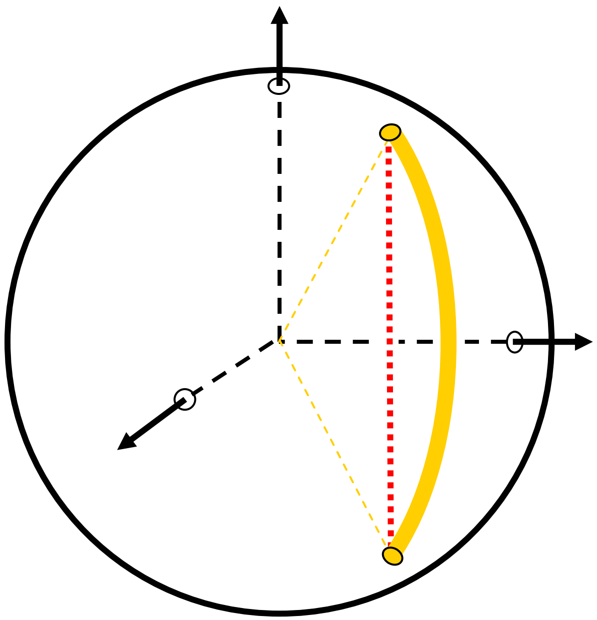
2 angles

- Messy because modulo 2π and pole

Use linear interpolation in 3D space

Orientation = projection onto the sphere

Same velocity correction



3 angles

Use the same principle

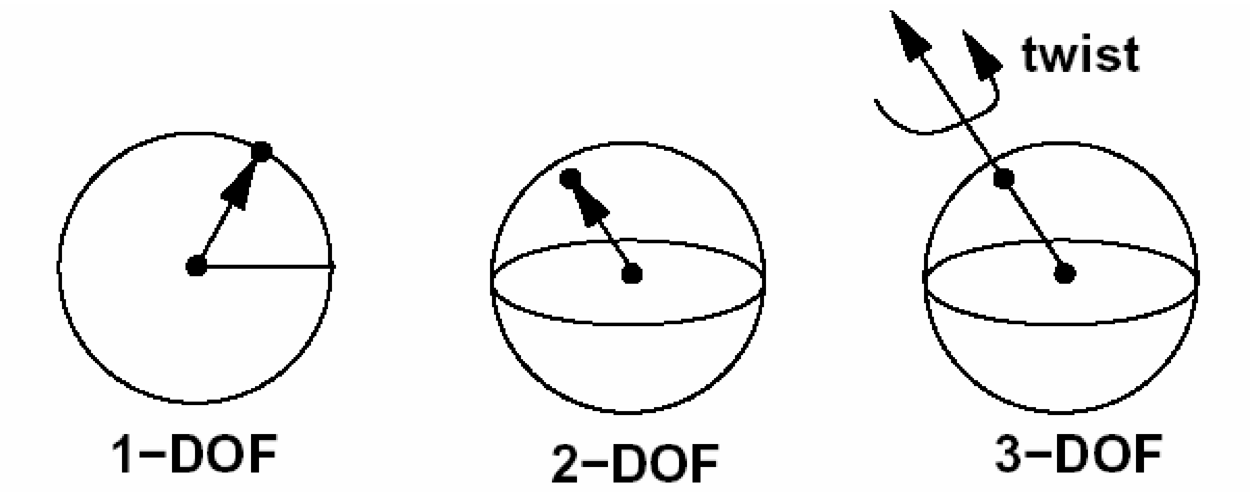
- interpolate in higher-dimensional space
- Project back to unit sphere

Probably need the 3-sphere embedded in 4D

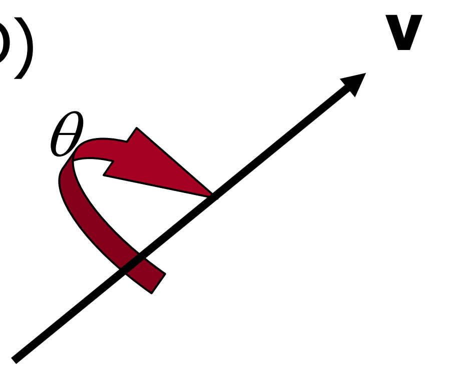
More complex, harder to visualize

Use the so-called Quaternions

- Due to Hamilton (1843); also Shoemake, Siggraph '85



Quaternions



Quaternions are unit vectors on 3-sphere (in 4D)

Right-hand rotation of θ radians about **v** is

$\mathbf{q} = \{\cos(\theta/2);\ \mathbf{v}\sin(\theta/2)\}$

- Often noted (s, **v**)
- What if we use –**v** ?

What is the quaternion of Identity rotation?

-

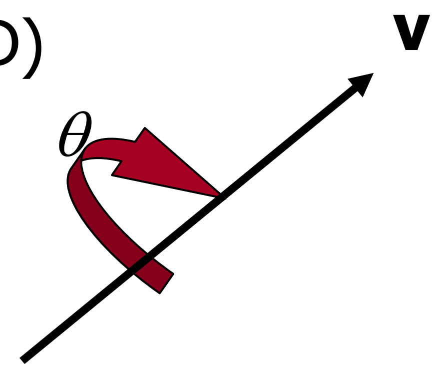
Is there exactly one quaternion per rotation?

-

What is the inverse $\mathbf{q}^{-1}$ of quaternion **q** {a, b, c, d}?

-

Quaternions



Quaternions are unit vectors on 3-sphere (in 4D)

Right-hand rotation of θ radians about **v** is

q = {cos($\theta/2$); **v** sin($\theta/2$)}

- Often noted (s, **v**)
- Also rotation of $-\theta$ around **-v**

What is the quaternion of Identity rotation?

- **q**i = {1; 0; 0; 0}

Is there exactly one quaternion per rotation?

- No, **q**={cos($\theta/2$); **v** sin($\theta/2$)} is the same rotation as **-q**={cos($(\theta+2\pi)/2$); **v** sin($(\theta+2\pi)/2$)}
- Antipodal on the quaternion sphere

What is the inverse $\mathbf{q}^{-1}$ of quaternion **q** {a, b, c, d}?

- {a, -b, -c, -d}

Quaternion Algebra

Two general quaternions are multiplied by a special rule:

$$\mathbf{q}_1\mathbf{q}_2 = \left(s_1 s_2 - \left(\vec{v}_1 \bullet \vec{v}_2\right), s_1\vec{v}_2 + s_2\vec{v}_1 + \vec{v}_1 \times \vec{v}_2\right)$$

Sanity check : {cos($\alpha/2$); **v** sin($\alpha/2$)} {cos($\beta/2$); **v** sin($\beta/2$)}

Quaternion Algebra

Two general quaternions are multiplied by a special rule:

$$\mathbf{q}_1\mathbf{q}_2 = \left(s_1 s_2 - \left(\vec{v}_1 \bullet \vec{v}_2\right), s_1\vec{v}_2 + s_2\vec{v}_1 + \vec{v}_1 \times \vec{v}_2\right)$$

Sanity check : {cos($\alpha/2$); **v** sin($\alpha/2$)} {cos($\beta/2$); **v** sin($\beta/2$)}

{cos($\alpha/2$)cos($\beta/2$) - sin($\alpha/2$)**v**. sin($\beta/2$)} **v**,
cos($\beta/2$) sin($\alpha/2$) **v** + cos($\alpha/2$)sin($\beta/2$) **v** +**v** $\times$ **v**}

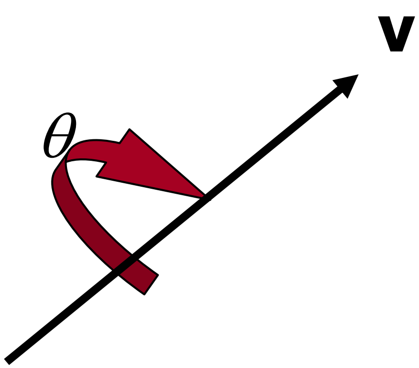
{cos($\alpha/2$)cos($\beta/2$) - sin($\alpha/2$) sin($\beta/2$),
v(cos($\beta/2$) sin($\alpha/2$) + cos($\alpha/2$) sin($\beta/2$))}

{cos($(\alpha+\beta)/2$), **v** sin($(\alpha+\beta)/2$) }

Quaternion recap 1 (wake up)

4D representation of orientation

$$\mathbf{q} = \{\cos(\theta/2);\ \mathbf{v}\sin(\theta/2)\}$$



Inverse is $\mathbf{q}^{-1} = (s, -\mathbf{v})$

Multiplication rule

$$\mathbf{q}_1\mathbf{q}_2 = \left(s_1 s_2 - \left(\vec{v}_1 \bullet \vec{v}_2\right), s_1\vec{v}_2 + s_2\vec{v}_1 + \vec{v}_1 \times \vec{v}_2\right)$$

- Consistent with rotation composition

How do we apply rotations?

How do we interpolate?

Quaternion Algebra

Two general quaternions are multiplied by a special rule:

$$\mathbf{q}_1\mathbf{q}_2 = \left(s_1 s_2 - \left(\vec{v}_1 \bullet \vec{v}_2\right), s_1\vec{v}_2 + s_2\vec{v}_1 + \vec{v}_1 \times \vec{v}_2\right)$$

To rotate 3D point/vector **p** by **q**, compute

- $\mathbf{q}\ \{0; \mathbf{p}\}\ \mathbf{q}^{-1}$

p= (x,y,z) **q**={ cos(θ/2), 0,0,sin(θ/2) } = {c, 0,0,s}

q {0,**p**}= {c, 0, 0, s} {0, x, y, z}

= {c.0- zs, c**p**+0(0,0,s)+ (0,0,s) $\times$ **p**}

= {-zs, c **p** + (-sy,sx,0) }

q {0,**p**} $\mathbf{q}^{-1}$ = {-zs, c **p** + (-sy,sx,0) } {c, 0,0,-s}

= {-zsc-(c**p**+(-sy,sx,0)).(0,0,-s),
-zs(0,0,-s)+c(c**p**+(-sy, sx,0))+ (c **p** + (-sy,sx,0)) x (0,0,-s) }

= {0, $(0,0,zs^2)+c^2\mathbf{p}+(-csy, csx,0)+(-csy, csx, 0)+(s^2x, s^2y, 0)$}

= {0, $(c^2x-2csy-s^2x, c^2y+2csx-s^2y, zs^2+sc^2)$}

= {0, x cos(θ)-y sin(θ), x sin(θ)+y cos(θ), z }

Quaternion Algebra

Two general quaternions are multiplied by a special rule:

$$\mathbf{q}_1\mathbf{q}_2 = \left(s_1 s_2 - \left(\vec{v}_1 \bullet \vec{v}_2 \right), s_1\vec{v}_2 + s_2\vec{v}_1 + \vec{v}_1 \times \vec{v}_2 \right)$$

To rotate 3D point/vector **p** by **q**, compute

- **q** {0; **p**} $\mathbf{q}^{-1}$

Quaternions are associative:

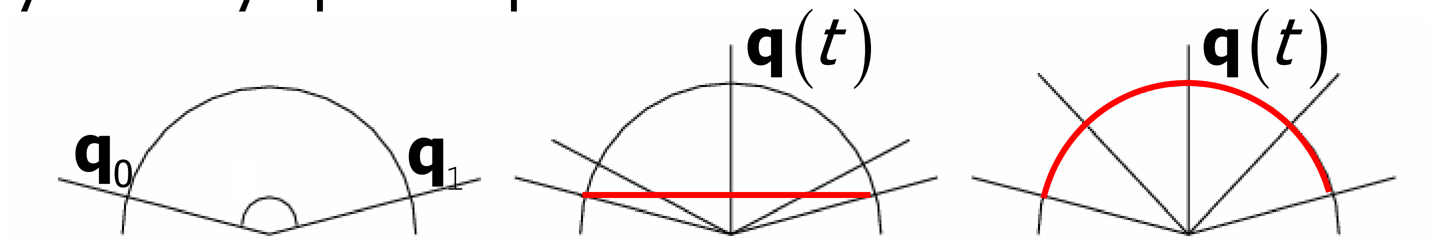
- (**q**1 **q**2) **q**3 = **q**1 (**q**2 **q**3)

But not commutative:

- **q**1 **q**2 $\neq$ **q**2 **q**1

Quaternion Interpolation (velocity)

The only problem with **l**inear int**erp**olation (lerp) of quaternions is that it interpolates the straight line (the secant) between the two quaternions and not their spherical distance. As a result, the interpolated motion does not have smooth velocity: it may speed up too much in some sections:



keyframes lerp slerp

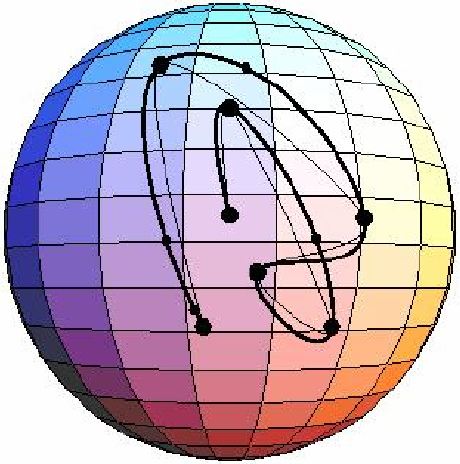
Spherical **l**inear int**erp**olation (slerp) removes this problem by interpolating along the arc lines instead of the secant lines.

$$\mathrm{slerp}(\mathbf{q}_0, \mathbf{q}_1, t) = \mathbf{q}(t) = \frac{\mathbf{q}_0 \sin\big((1-t)\omega\big) + \mathbf{q}_1 \sin(t\omega)}{\sin(\omega)},$$

where $\omega = \cos^{-1}(\mathbf{q}_0 \cdot \mathbf{q}_1)$

Quaternion Interpolation

Higher-order interpolations: must stay on sphere



See Shoemake paper for:

- Matrix equivalent of composition
- Details of higher-order interpolation
- More of underlying theory

Problems

- No notion of favored direction (e.g. up for camera)
- No notion of multiple rotations, needs more key points

Quaternions

Can also be defined like complex numbers

$a+bi+cj+dk$

Multiplication rules

- $i^2=j^2=k^2=-1$
- $ij=k=-ji$
- $jk=i=-kj$
- $ki=j=-ik$

…

Fun:Julia Sets in Quaternion space

Mandelbrot set: $Z_{n+1}=Z_n^2+Z_0$

Julia set $Z_{n+1}=Z_n^2+C$

http://aleph0.clarku.edu/~djoyce/julia/explorer.html

Do the same with Quaternions!

Rendered by Skal (Pascal Massimino) http://skal.planet-d.net/

Images removed due to copyright considerations.

See also http://www.chaospro.de/gallery/gallery.php?cat=Anim

Fun:Julia Sets in Quaternion space

Julia set $Z_{n+1}=Z_n^2+C$

Do the same with Quaternions!

Rendered by Skal (Pascal Massimino) http://skal.planet-d.net/

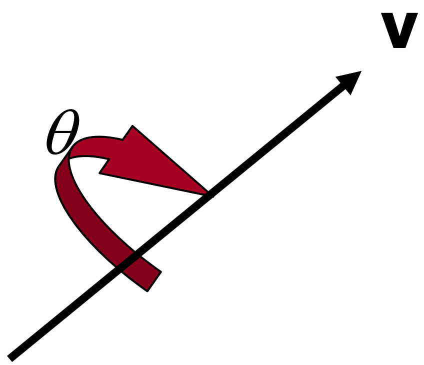
This is 4D, so we need the time dimension as well

Images removed due to copyright considerations.

Recap: quaternions

3 angles represented in 4D

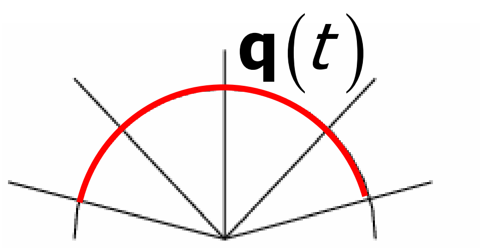
$$\mathbf{q} = \{\cos(\theta/2);\ \mathbf{v}\sin(\theta/2)\}$$



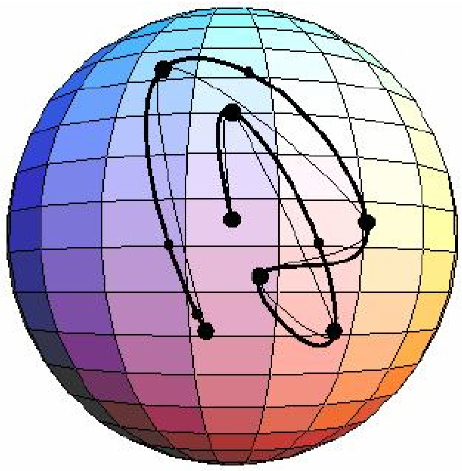
Weird multiplication rules

$$\mathbf{q}_1\mathbf{q}_2 = \left(s_1 s_2 - \left(\vec{v}_1 \bullet \vec{v}_2\right),\ s_1\vec{v}_2 + s_2\vec{v}_1 + \vec{v}_1 \times \vec{v}_2\right)$$

Good interpolation using slerp



slerp



Overview

Interpolation of rotations, quaternions

- Euler angles
- Quaternions

Dynamics

- Particles
- Rigid body
- Deformable objects

Break: movie time

Pixar *For the Bird*